SDI – Sistemas Distribuidos e Internet

ENUNCIADO PRÁCTICA 1 – SPRING

INFORME

Grupo 605-614

Nombre1:	Pelayo
Apellidos1:	García Álvarez
Email1:	UO264761@uniovi.es
Cód. ID GIT	605
Participación	50%-50%
Nombre1:	Adrián
Apellidos1:	Pérez Manso
Email1:	UO265081@uniovi.es
Cód. ID GIT	614

Índice

Introducción

El objetivo principal de la realización de este proyecto es desarrollar una aplicación Web utilizando Spring, un *framework* basado en *JEE* que usa el patrón *MVC* cuya característica principal es el uso de un modelo *POJO*. Así mismo la aplicación será desarrollada utilizando *Spring Boot*, con el objetivo de facilitar su desarrollo (configuración, generación de código, servidor embebido, …).

Nuestra aplicación en concreto se trata de una de tipo red social en el cual existirán usuarios los cuales pueden ser de tres tipos:

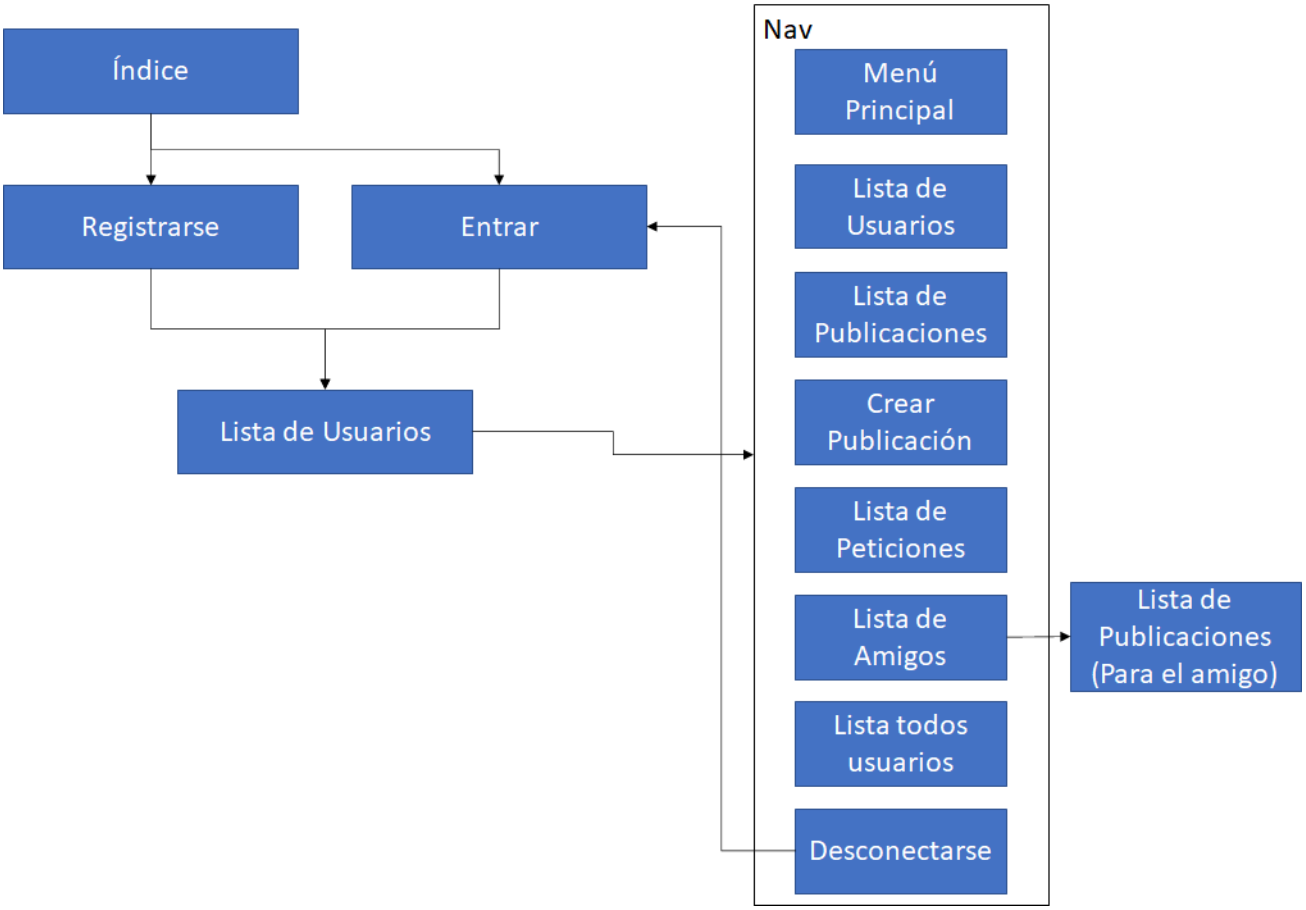
- Publico (Anónimo)
- Usuario Administrador
- Usuario Estándar

En cuanto a las funcionalidades que tendrán disponibles cada uno de los distintos usuarios se explicará con más detalle en el punto de aspectos técnicos y de diseño relevantes.

Mapa de navegación

El mapa de navegación es el que se muestra a continuación. Es importante añadir que todas las opciones accesibles que se encuentran en el *nav* son accesibles desde cualquier parte de la aplicación, solo una vez que estás dentro.

Además, cabe destacar que la lista de todos los usuarios solo es visible y accesible si eres administrador. Así mismo existe una opción en el *nav* de los administradores que no incluimos en el mapa porque añadiría complejidad al esquema y lo único que hace es redirigir a la ventana menú principal, esta opción se utilizó para testear la ultima prueba de la parte de seguridad, pero en cuanto a la funcionalidad es irrelevante, por lo tanto, lo hemos omitido del diagrama.



Aspectos técnicos y de diseño relevantes

En este apartado vamos a explicar el enfoque de nuestras soluciones a los diferentes requisitos que merecen ser mencionados y son más complejos.

Lista de usuarios

La lista de usuarios a la que se accede cada vez que entras a la aplicación o te registras en ella muestra todos los usuarios estándar excepto el usuario con el que has accedido, ya que no le vemos sentido que este aparezca (enviarte una solicitud de amistad a ti mismo). Además, no se muestran los usuarios administradores de la aplicación, esto es debido principalmente a que aparece como requisito en el cuarto punto del enunciado del problema, pero nosotros los añadiríamos a la lista ya que estos puedes realizar las mismas acciones que uno estándar, gracias a que la aplicación es bastante flexible y con una arquitectura ordenada si en un futuro quisiéramos cambiar esta funcionalidad simplemente habría que llamar desde el controlador a un método distinto del servicio que devuelva la lista de todos los usuarios y no habría que tocar nada del resto del código.

Sin embargo, los administradores tienen acceso a ver todos los usuarios tanto estándar como administradores desde la ventana todos los usuarios. Ya que desde esta ventana un administrador debe ser capaz de gestionar todos los usuarios.

Lista de publicaciones

En esta vista se muestran únicamente las publicaciones del propio usuario que ha creado con anterioridad. Si este quiere ver las publicaciones de otros usuarios primero debe agregarlos como amigos y después desde el listado de amigos acceder a las publicaciones de ese usuario. Todo esto es gracias a que nosotros vemos los amigos como una relación entre usuarios, y cada uno tiene un conjunto de publicaciones.

Es importante señalar que todos los formularios tienen validación en el servidor y se ha creado una página *HTML* llamada 403 que es utilizada para sobrescribir justamente la ventana que viene por defecto, para cuando un usuario estándar quiere acceder, modificando la URL, a una acción/ventana exclusiva de los administradores.

Sistema de amistad

La vista “users/index.html” (que usa el “fragmento fragments/usersList.html”) muestra al lado de cada usuario listado información sobre el estado de la relación de este con el usuario autenticado. Si aparece el botón "Añadir como amigo", el usuario que tiene la sesión podrá enviar una petición al que le corresponda el botón. Esta petición es manejada por el controlador, que llama al servicio correspondiente, que a su vez accede al repositorio, para que cree una nueva instancia de *FriendPetition*, la cual guarda quién ha enviado la petición (*sendUser*), quién la recibe (*arriveUser*) y si ha sido aceptada o no (*hasAccepted*), sirviendo este último campo para diferenciar posteriormente entre "sois amigos" o "petición enviada". La vista mostraría entonces un mensaje en el lugar del botón (Petición enviada). Si el usuario que recibe la petición la acepta, otro controlador se encargará de invocar al servicio que modificará el booleano en la entrada en la base de datos que contenga al usuario autenticado como receptor y al primero como remitente. Al navegar a la vista de usuarios, ahora el texto que aparece es "Sois amigos".

Publicaciones con imagen adjunta

Un campo "<input type='file'/>" recoge el archivo que se corresponderá con la foto de la publicación, el cual acepta por defecto archivos *.png*, *.jpg* y *.jpeg*, aunque dado el Explorador de archivos de Windows se pueden elegir otros archivos, por lo que se han tomado otras medidas para evitarlo, las cuales se explicaran más adelante.

El *enctype* del formulario es, a partir de este punto, "multipart/form-data" lo que permite añadirle al controlador un parámetro *MultipartFile* para manejar el archivo, el cual será guardado en la entidad *Publication* (pero no guardado en la base de datos) para ser validado. El controlador decide en función de si el archivo está vacío o no si es una publicación sin imagen o no. La validación consiste, además de la estándar, en comprobar que el archivo adjuntado cumple con las extensiones anteriormente mencionadas.

Borrado múltiple de usuarios

Cada *checkbox* tiene como valor asignado el id del usuario al que corresponde, por lo que al pasar la lista de *checkboxes* solo se ha de acceder por cada una de ellas al valor que contienen y pasar ese id al servicio para que borre el usuario correspondiente, eliminando así publicaciones y amistades relacionadas.

Validación del *login*

Se pasan a la vista dos atributos vacíos, para el *logout* y los errores respectivamente. En cualquiera de los casos, un parámetro se añade a la URL indicando el comportamiento, por lo que el objeto pasado a las vistas a través del modelo dejaría de ser nulo. En la vista, a su vez, se comprueba que estos objetos no sean nulos, y si es así, el mensaje correspondiente aparecerá en pantalla.

Información necesaria para el despliegue y ejecución

Ejecución

Para la correcta ejecución de la aplicación simplemente es necesario iniciar previamente la base de datos *HSQLDB*, no siendo necesario ejecutar la que nosotros subimos ya que la aplicación crea la base de datos cada vez que se inicia. Así mismo es posible que sea necesario cambiar en las propiedades de los archivos *messages_x.properties* para que estén en UTF-8, así como el workspace.

Pruebas

Primero es necesario construir la ruta de *Selenium* que se encuentra en la carpeta *lib*. Una vez hecho esto es importante poner en la ruta de *Geckdriver024* y del ejecutable de *Firefox.*

Por último, es necesario parar la ejecución de la aplicación y volver a iniciarla para que se reinicie la base de datos y las pruebas pasen sin problemas.

Conclusión

Como conclusión podríamos añadir que la práctica ha resultado bastante útil, tanto para si en un futuro queremos crear nuestra propia pagina web de una forma sencilla, como para reforzar los conocimientos que hemos ido adquiriendo a lo largo de las distintas entregas de las prácticas de esta primera parte de la asignatura. Además, hay funcionalidades que tuvimos que hacer si queríamos completar todos los puntos, tanto obligatorios como opcionales, que requirieron buscar por internet información al respecto, lo cual nos hizo aprender más y conocer mejor el ecosistema de *Sprint Boot*.

Como último apunte, nos hubiera gustado profundizar más en la aplicación y que hubiera requerido cierto componente más creativo, no cumplir unos requisitos estrictos sino, por ejemplo, que un requisito fuera que la pagina tuviera un listado, página de registro página de login… Pero que la temática fuera a nuestra elección. Esta es nuestra opinión, además comprendemos que sería mucho mas complicado y el tiempo no es algo de lo que gocemos.